AUEB M.S.c in Data Science (part-time)

Text Analytics: Homework 2: Report

Team members:

Kaplanis Alexandros (P3351802)

Politis Spiros (P3351814)

Proimakis Manos (P3351815)

1. SETUP & EXECUTION

Please refer to the README.md for instructions.

2. INTRODUCTION

For the second assignment we decided to perform a sentiment analysis on tweets. The dataset was downloaded from Kaggle for the Sentiment140 project and the bundle contained a training dataset with 1.6 million Tweets (cleared from emoticons) and a test dataset.

The datasets have 2 classes; 0 for negative and 4 for positive.

The classes in the training dataset are equiprobable with 800,000 records per class.

2.1 DATA CLEANSING & PREPROCESSING

A lot of emphasis was given on the data preprocessing phase during which we created the **Preprocess** class with which we removed all usernames (words beginning with @) and all the hyperlinks from the dataset. Concluding the preprocess phase, we dropped all duplicate instances per user and kept only the useful columns (polarity, text).

2.2 DATA MANIPULATION

As a next step we **split** the Training dataset to train and test using SKlearn's *train_test_split* with a ratio ¾. This allowed us to keep the other test dataset that came with the bundle as **unseen**, in order to able to use it in the final stage of the assignment.

Additionally, we made the decision of keeping sample of 50,000 records (25,000 per class due to the class equiprobability).

As for the vectorization we proceeded with examining a couple of approaches on it:

a) Perform feature extraction using the TF-IDF vectorizer which was also used for feature selection using its *max_features* parameter and selecting 15,000 features.

   This first dimensions reduction was mostly done to reduce the running times.

b) Use a Twitter-corpus, pre-trained Word2Vec model ("glove-twitter-200"). We acquired the model using the built-in downloading functionality of the GenSim package, however it can also be downloaded from the official Stanford repo: https://nlp.stanford.edu/projects/glove/

   This model was selected based on the notion that, having been trained on tweets, it would perform good on our dataset. We opted for the 200-dim vectors model, assuming that it would provide more information, hence lead to better classifier performance.

To our surprise, the Word2Vec model did not provide significant gains in classification tasks, being only to provide slight improvement for some classifiers (e.g. kNN) in f-1 score, while performing worse in others (e.g. SVM, Logistic Regression).

3. CLASSIFIERS

It is worth mentioning that due to the long execution times of the classifiers we created a function that saves the trained models and also loads them when they need to be used.

For each classifier we have output a number of evaluation measures such as Precision, Recall, F1-scores, confusion matrices. In addition, we have also calculated the scores' macro and weighted averages and plotted the Precision-Recall and Learning curves. Our go-to evaluation measures are mostly F1-scores, the confusion matrices and the curves.

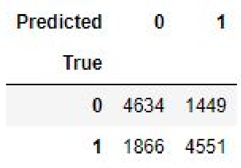
Note: In this assignment we assume that misclassifying a Negative (0) tweet is worse than a misclassifying a Positive (1) one.

For the training of the models we used 25,000 features and the following classifiers:

Dummy Classifier: It was used as a reference point which is validated by the equiprobability of the classes of the training dataset.

Naive Bayes: It could only be trained with the TF-IDF vectors since Word2Vec outputs negative values as well. Though it works best with integers, it is stated in the available documentation that it also works with TF-IDF values.

The F1-Scores were (apx.) 0.83 and 0.73 for the train and test data respectively. In addition, we had about 3 times more True Positives and Negatives than the False ones. ***(the indices are 0 for Negatives and 1 for Positives)***



Predicted	0	1
True		
0	4634	1449
1	1866	4551

Logistic Regression: It was trained with both TF-IDF vectors and Word2Vec. Both the test and train F1-Scores are better with the TF-IDF vectors.

Train F1-Scores: TF-IDF: 0.82 - Word2Vec: 0.74

Test F1-Scores: TF-IDF: 0.74 - Word2Vec: 0.73

Comparing the two classifiers, on one hand we have an increase in the test F1-Score, albeit small, and a decrease in the False Positives. On the other, we observe an increase in the False Negatives. As a result we cannot conclude that the Logistic Regression performed better than Naive Bayes.

TF-IDF

Predicted	0	1
True		
0	4517	1566
1	1582	4835

Word2Vec

Predicted	0	1
True		
0	4576	1657
1	1706	4561

However, it is worth mentioning that the adequate effectiveness of the Logistic Regression could be an indicator of no multicollinearity between the features as it is assumed by the classifier.

SVM: We trained this classifier with and without dimensionality reduction using both the TF-IDF vectors and the Word2Vec. The dimensionality reduction was achieved through the Truncated SVD method using 100 components.

The pre-reduced SVM scored almost identical train and test F1s with the Logistic Regression on both the TF-IDF and Word2Vec respectively. These high F1-Scores encouraged us that the post-reduced SVM would score higher. However, the use of the TF-IDF vectors brought a 6 point reduction to the test F1-score and a 1 point decrease to the test F1 of the WordVec.

Before Dimensional Reduction

TF-IDF

Predicted	0	1
True		
0	4565	1668
1	1520	4747

Word2Vec

Predicted	0	1
True		
0	4554	1679
1	1680	4587

Post Dimensional Reduction

TF-IDF

Predicted	0	1
True		
0	4114	1969
1	1998	4419

Word2Vec

Predicted	0	1
True		
0	4555	1678
1	1732	4535

A first suspect for this is that we reduced the dimension more than it was needed crowding the data in one spot, making them harder to be linearly classified but this should also occur with the Word2Vec as well. In the end however, the fact that the SVM classifier costs O(features×observations^2) to run deterred us from experimenting further.

KNN: The last classifier used was kNN. We used k = 5 for a first attempt which yielded the rather disappointing test F1-Scores of 0.58 and 0.65 for the TF-IDF and Word2Vec respectively. Due to long execution times we decided to tune the parameter k in the optimization phase.

TF-IDF

Predicted	0	1
True		
0	2901	3182
1	2099	4318

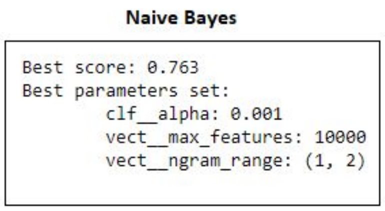
Word2Vec

Predicted	0	1
True		
0	3930	2303
1	2111	4156

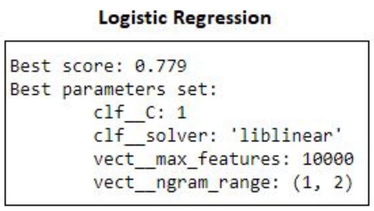
3.1 OPTIMIZATION & PARAMETER TUNING

For the optimization phase we used SKlearn's GridSearchCV function along with a 5-fold cross validation. Once again the SVM presented issues with the execution times, so it was excluded from the optimization.

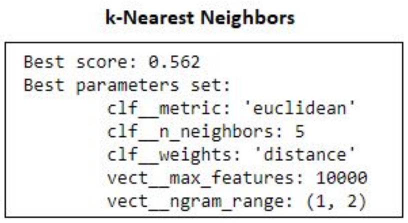
During the optimization phase, KNN didn't score any better and Naive Bayes gave a slightly better score. The best score was given ultimately by the Logistic Regression classifier.


Naive Bayes

```
Best score: 0.763
Best parameters set:
        clf__alpha: 0.001
        vect__max_features: 10000
        vect__ngram_range: (1, 2)
```

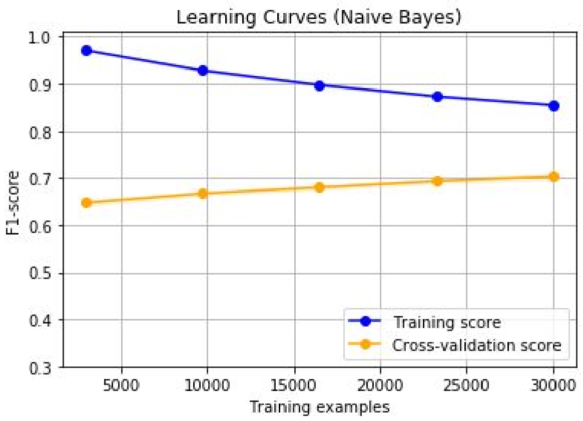

Logistic Regression

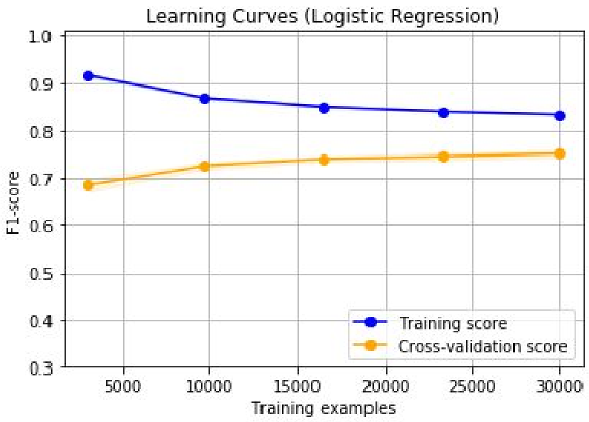
```
Best score: 0.779
Best parameters set:
        clf__C: 1
        clf__solver: 'liblinear'
        vect__max_features: 10000
        vect__ngram_range: (1, 2)
```


k-Nearest Neighbors

```
Best score: 0.562
Best parameters set:
        clf__metric: 'euclidean'
        clf__n_neighbors: 5
        clf__weights: 'distance'
        vect__max_features: 10000
        vect__ngram_range: (1, 2)
```

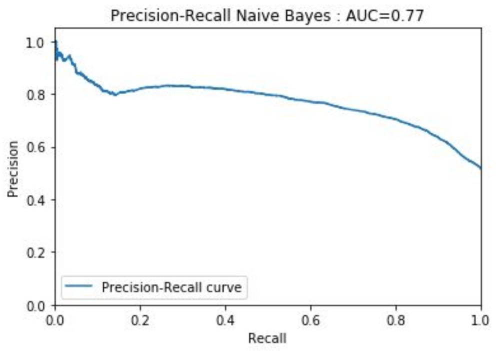
But still it wasn't entirely clear, which classifier to select between Naive Bayes and Logistic Regression. So, we decided to use the Learning and the Precision-Recall curves in order to decide which one to select.

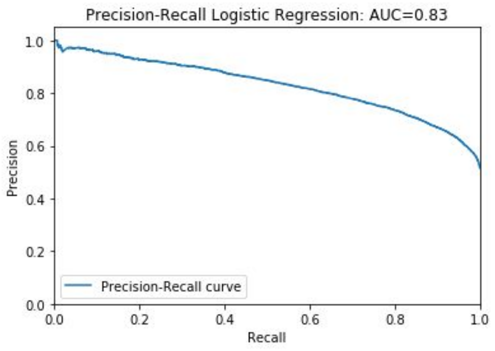




The Learning Curves show that we could certainly use more than 25,000 records to train the classifiers (especially NB) since the curves in both diagrams are not close to converging. Maybe 50 or 70 thousand records would suffice. However, both cross-validation curves are similarly smooth with their respective training curves, meaning that the error does not suffer because of the variance. The fact that NB needs more data than the Logistic Regression for the learning curves to converge gives an edge to the latter.

As a last evaluation measure we used Precision-Recall curves. Though our data are balanced and we could use ROC curves, we decided that PRC were a safer choice.





Based on the above Precision-Recall curves we can observe that the AUC of the Logistic Regression is better than that of the Naive Bayes and as a result it seals the deal as the classifier of our choice.

4. RESULTS

Before using the **unseen** dataset we normalized it, meaning that we applied the same preprocess as with the training data. The resulting F1-Score was 0.77 and he confusion matrix of the unseen dataset was the below:

Predicted	0	4
True		
0	122	55
4	27	155

The final classification gives a good ratio between the TN-FN (5/2) and TP-FP (5/1) so we have an overall efficient classifier. Some random examples: *(the notebook has more examples of mismatches)*

True	Pred	Tweet
POSITIVE	POSITIVE	just got back from the movies went to see the new night at the museum with rachel it was good
POSITIVE	POSITIVE	awesome viral marketing for funny people teach
NEGATIVE	NEGATIVE	nancy pelosi gave the worst commencement speech i ve ever heard yes i m still bitter about this
NEGATIVE	POSITIVE	i m itchy and miserable